We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

A New Hybrid Particle Swarm Optimization Algorithm to the Cyclic Multiple-Part Type Three-Machine Robotic Cell Problem

Isa Nakhai Kamalabadi, Ali Hossein Mirzaei and Saeede Gholami
Department of Industrial Engineering, Faculty of Engineering, Tarbiat Modares University
Tehran, Iran

1. Introduction

Nowadays the level of automation in manufacturing industries has been increased dramatically. Some examples of these automation progresses are in cellular manufacturing and robotic cells. A growing body of evidence suggests that, in a wide variety of industrial settings, material handling within a cell can be accomplished very efficiently by employing robots (see (Asfahl, 1992)). Among the interrelated issues to be considered in using robotic cells are their designs, the scheduling of robot moves, and the sequencing of parts to be produced.

Robotic cell problem in which robot is used as material handling system received considerable attentions. Sethi et al. (1992) proved that in buffer-less single-gripper two-machine robotic cells producing single part-type and having identical robot travel times between adjacent machines and identical load/unload times, a 1-unit cycle provides the minimum per unit cycle time in the class of all solutions, cyclic or otherwise. For three machine case, Crama and van de Klundert (1999), and Brauner and Finke (1999) shown that the best 1-unit cycle is optimal solution for the class of all cyclic solutions. Hall et al. (1997; 1998) considered the computational complexity of the multiple-type parts three-machine robotic cell problem under various robot movement policies. This problem is studied for no-wait robotic cells too. For example Agnetis (2000) found an optimal part schedule for no-wait robotic cells with three and two machines. Agnetis and pacciarelli (2000) have studied partscheduling problem for no-wait robotic cells, and found the complexity of the problem. Crama et al. (2000) studied flow-shop scheduling problems, models for such problems, and complexity of theses problems. Dawande et al. (2005) reviewed the recent developments in robotic cells and, provided a classification scheme for robotic cells scheduling problem. Some other special cases have been studied such as: Drobouchevitch et al. (2006) provided a model for cyclic production in a dual-gripper robotic cell. Gultekin et al. (2006) studied robotic cell scheduling problem with tooling constraints for a two-machine robotic cell where some operations can only be processed on the first machine and some others can only be processed on the second machine and the remaining can be processed on both machines.

Gultekin et al. (2007) considered a flexible manufacturing robotic cell with identical parts in which machines are able to do different operations and the operation time is not system parameter and is variable. They proposed a lower bound for 1-unit cycles and 2-unit cycles. Sriskandarajah et al. (1998) classified the part sequence problems associated with different robot movement policies, in this chapter a robot movement policy is considered, which its part scheduling problem is NP-Hard, and Baghchi et al. (2006) proposed to solve this problem, by a heuristic or meta-heuristic. In this chapter a meta-heuristic method based on particle swarm optimization is applied to solve the problem.

In this chapter an m-machine flexible cyclic cell is considered. All parts in an MPS (A minimal part set) visit each machine in the same order, the production environment is cyclic, and parts are produced at the same order repeatedly.

In this chapter, we consider multiple-type parts three-machine robotic cells which have operational flexibility in which the operations can be performed in any order; moreover each machine can be configured to perform any operation. To explain the problem, consider a machining centre where three machine tools are located and a robot is used to feed the machines namely M_1, M_2, M_3 (see figure 1). All parts are brought to and removed from the robotic cell by Automated Storage & Retrieval System (AS/RS). The pallets and feeders of the AS/RS system allow hundreds of parts to be loaded into the cell without human intervention. The machines can be configured to perform any operation.

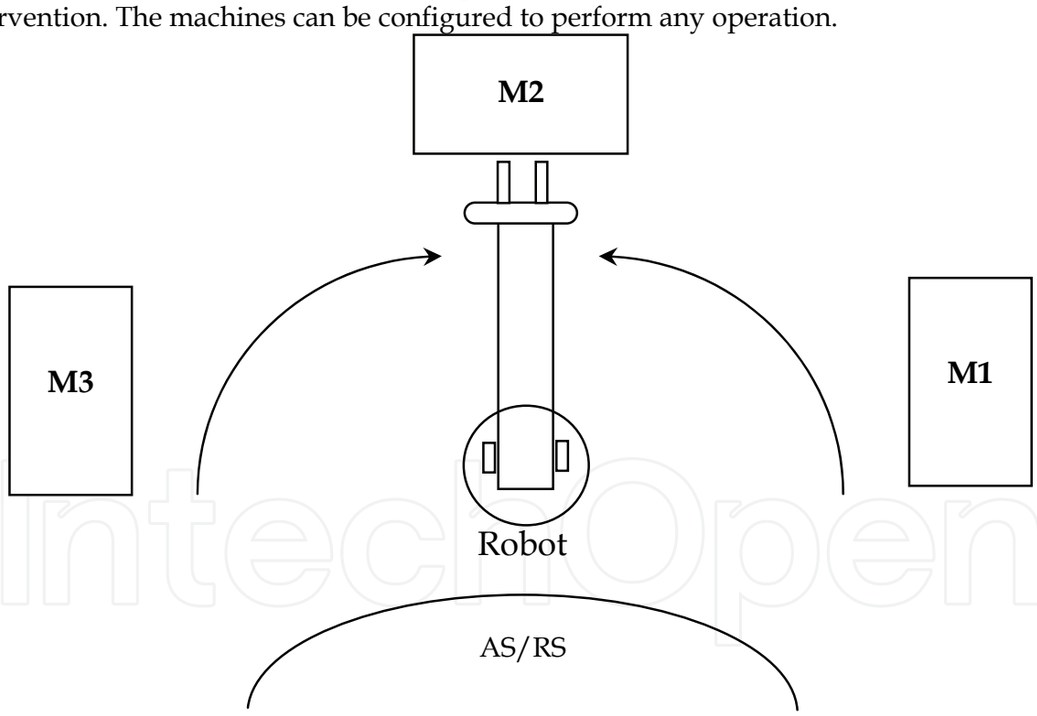


Fig. 1. Robotic work cell layout with three machines

The aim of this chapter is to find a schedule for the robot movement and the sequence of parts to maximize throughput (i.e., to minimize cycle time), as it is showed that this problem is NP-Complete in general (see Hall et al. (1997)). Hence, this chapter proposes a novel hybrid particle swarm optimization (HPSO) algorithm to tackle the problem. To validate the developed model and solution algorithm, various test problems with different sizes is

randomly generated and the performance of the HPSO is compared with three benchmark metaheuristics: Genetic Algorithm, PSO-I (basic Particle Swarm Optimization algorithm), and PSO-II (constriction Particle Swarm Optimization algorithm). The rest of this chapter is organized as follows: The problem definition and required notations are presented in Section 2, Section 3 presents the developed mathematical model, and in Section 4, the proposed hybrid particle swarm optimization algorithm is described. The computational results are reported in Section 5, and the conclusions are presented in Section 6.

2. Problem definition

The robotic cell problem is a special case of the cyclic blocking flow-shop, where the jobs might block either the machine or the robot. In a cyclic schedule the same sequences repeat over and over and the state of the cell at the beginning of each cycle is the similar to the next cycle. It is assumed that the discipline for the movements of parts is an ordinary flow-shop discipline. That is a part meets machines M_1, M_2, M_3 consequently.

2.1 Notations

The following notation is used to describe the robotic cell problem:

m : The number of machines

I/O : The automated input-output system for the cell

PT_i : The part-type i to be produced

r_i : The minimal ratio of part i to be produced

MPS : The number of part set consisting r_i parts of type PT_i

n : the total number of parts to be produced in the MPS ($n = r_1 + r_2 + \ldots + r_k$)

a_i : The processing time of part i on M_1

b_i : The processing time of part i on M_2

c_i : The processing time of part i on M_3

δ : Robot travelling time between two successive machines (I/O is assumed as machine M_0)

ε : The load/unload time of part i

w_i^j : The robot waiting time on M_j to unload part i

S^k : The robot movement policy S under category k

T^k : The cycle time under S^k

In this study the standard classification scheme for scheduling problems: $\psi_1 | \psi_2 | \psi_3$ is used where ψ_1 indicates the scheduling environment, ψ_2 describes the job characteristics and ψ_3 defines the objective function (Dawande et al., 2005). For example

$FRC_3 \mid k \geq 2, S^1 \mid C_t$ denotes the minimization of cycle time for multi-type part problem in a three flow-shop robotic cell, restricted to robot move cycle S^1.

2.2 Three machine robotic flow shop cell $FRC_3 \mid K \geq 2 \mid C_t$

In the three machine robotic flow shop cell, there are six different potentially optimal policies for robot to move the parts between the machines (Bagchi et al., 2006). Sethi et al. (1992) showed that any potentially optimal one-unit robot move cycle in a m machine robotic cell can be described by exactly m+1 following basic activities:

M_i^- : Load a part on M_i $\quad i = 1, 2, \ldots, m$

M_i^+ : Unload a finished part from M_i $\quad i = 1, 2, \ldots, m$

In other words, a cycle can be uniquely described by a permutation of the m+1 activity. The following are the available robot move cycles for m=3 flow-shop robotic cell (Sethi et al., 1992):

$$S^1 : \left\{M_3^+, M_1^-, M_2^-, M_3^-, M_3^+\right\}$$

$$S^2 : \left\{M_3^+, M_1^-, M_3^-, M_2^-, M_3^+\right\}$$

$$S^3 : \left\{M_3^+, M_3^-, M_1^-, M_2^-, M_3^+\right\}$$

$$S^4 : \left\{M_3^+, M_2^-, M_3^-, M_1^-, M_3^+\right\}$$

$$S^5 : \left\{M_3^+, M_2^-, M_1^-, M_3^-, M_3^+\right\}$$

$$S^6 : \left\{M_3^+, M_3^-, M_2^-, M_1^-, M_3^+\right\}$$

In this chapter we consider a three machine robotic cell problem under the S^6 policy (Figure 2). The problem of finding the best part sequence using the robot move cycle S^6 is NP-complete (Hall et al., 1998).

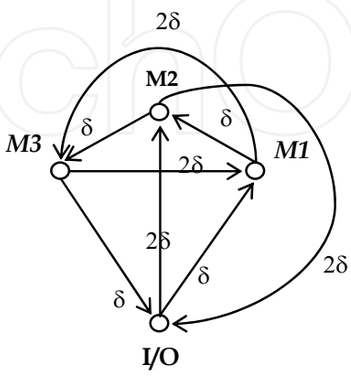


Fig. 2. The robot movement under S^6

Lemma 1. The cycle times of one unit for the policy s^6 are given by:

$$\mathrm{T}^6_{\mathrm{I},\sigma(\mathrm{i})\sigma(\mathrm{i}+1)\sigma(\mathrm{i}+2)} = 12\delta + 8\varepsilon + \max\{0, \mathrm{a}_{\sigma(\mathrm{i}+2)}\text{-}8\delta\text{-}4\varepsilon, \mathrm{b}_{\sigma(\mathrm{i}+1)}\text{-}8\delta\text{-}4\varepsilon, \mathrm{c}_{\sigma(\mathrm{i})}\text{-}8\delta\text{-}4\varepsilon\}$$

Proof: According to figure 2 the robot movement under policy s^6 is as follow:

Pickup part p_{i+2} from I/O(ε) move it to $\mathrm{M}_1(\delta)$ load p_{i+2} onto $\mathrm{M}_1(\varepsilon)$ go to $\mathrm{M}_3(2\delta)$ if necessary wait at $\mathrm{M}_3(\mathrm{w}^{\mathrm{i}}{}_3)$, unload p_i from $\mathrm{M}_3(\varepsilon)$ move it to I/O(δ) drop p_i at I/O(ε) go to $\mathrm{M}_2(2\delta)$ if necessary wait at $\mathrm{M}_2(\mathrm{w}^{\mathrm{i+1}}{}_2)$, unload p_{i+1} from $\mathrm{M}_2(\varepsilon)$ move it to $\mathrm{M}_3(\delta)$, load P_{i+1} onto $\mathrm{M}_3(\varepsilon)$ go to $\mathrm{M}_1(2\delta)$ if necessary wait at $\mathrm{M}_1(\mathrm{w}^{\mathrm{i+2}}{}_1)$, unload P_{i+2} from $\mathrm{M}_1(\varepsilon)$ move it to $\mathrm{M}_2(\delta)$ load P_{i+2} onto $\mathrm{M}_2(\varepsilon)$ go to I/O(2δ) then start a new cycle by picking up the part P_{i+3}.

The cycle time by considering waiting times is as follow:

$$T^6_{I,\sigma(i)\sigma(i+1)\sigma(i+2)} = 12\delta + 8\varepsilon + w_1^{i+2} + w_2^{i+1} + w_3^i$$

$$w_1^{i+2} = \max\{0, a_{\sigma(i+2)} - w_2^{i+1} - w_3^i - 8\delta - 4\varepsilon\}$$

$$w_2^{i+1} = \max\{0, b_{\sigma(i+1)} - w_3^i - 8\delta - 4\varepsilon\}$$

$$w_3^i = \max\{0, c_{\sigma(i)} - w_1^{i+2} - 8\delta - 4\varepsilon\}$$

$$T^6_{I,\sigma(i)\sigma(i+1)\sigma(i+2)} = 12\delta + 8\varepsilon + \max\{0, a_{\sigma(i+2)} - 8\delta - 4\varepsilon, b_{\sigma(i+1)} - 8\delta - 4\varepsilon, c_{\sigma(i)} - 8\delta - 4\varepsilon\}$$

3. Developing mathematical model

In this section we develop a systematic method to produce necessary mathematical programming formulation for robotic cells. Therefore first we model single-part type problem through Petri nets, and then extend the model to multiple-part type problem.

A Petri-net is a four-tuple $PN(P,T,A,W)$, where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, $T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs, and $W : A \to \{1, 2, 3, ...\}$ is a weight function.

Every place has an initial marking $M_0 : P \to \{0, 1, 2, ...\}$. If we assign time to the transitions we call it as Timed Petri net.

The behaviour of many systems can be described by system states and their changes, to simulate the dynamic behaviour of system; marking in a Petri-net is changed according to the following transition (firing) rule: 1) A transition is said to be enabled if each input place *p* of *t* is marked at least with $w(p,t)$ tokens, where $w(p,t)$ is weight of the arc from *p* to *t*. 2) An enabled transition may or may not be fired (depending on whether or not the event takes place). A firing of an enabled transition *t* removes $w(p,t)$ tokens from each input place *p* of *t* and adds $w(p,t)$ tokens to each output place *p* of *t*, where $w(p,t)$ is the weight of the arc from *t* to *p*.

By considering a single-part type system, the robot arm at steady state is located at machine M_2, therefore by coming back to this node we have a complete cycle for the robot arm.

The related Petri net for robot movements is shown in Figure 3 and the descriptions of the nodes for this graph with respective execution times would be as follows:

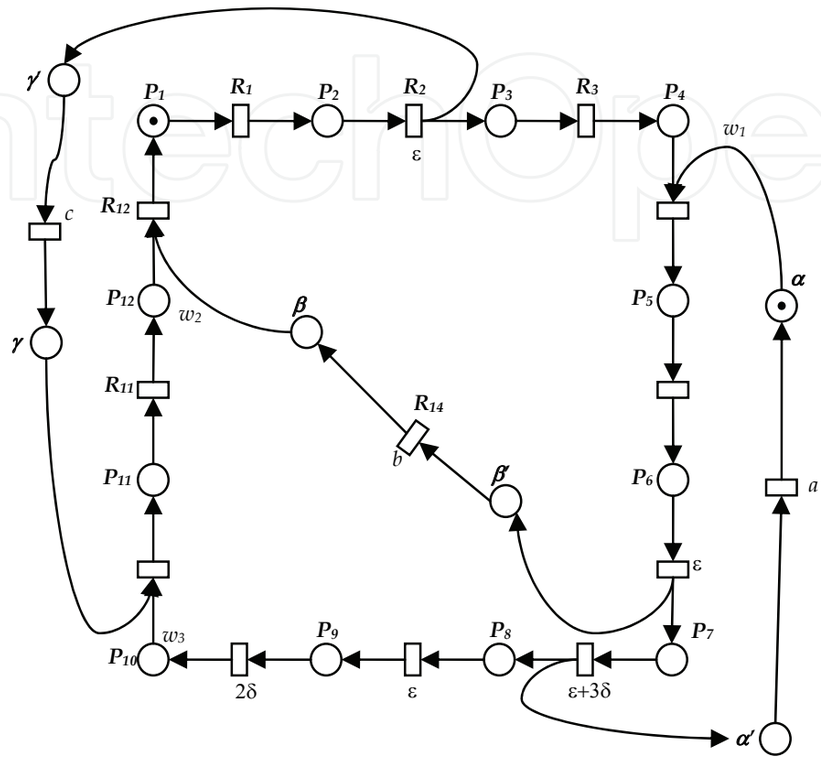


Fig. 3. Petri net for s^6 policy

R_1: go to $M_3(\delta)$; R_2: load $M_3(\varepsilon)$; R_3: go to $M_1(2\delta)$;

R_4: unload $M_1(\varepsilon)$; R_5: go to $M_2(2\delta)$; R_6: load $M_2(\varepsilon)$;

R_7: go to input, pickup a new part, go to $M_1(\varepsilon+3\delta)$; R_8: load $M_1(\varepsilon)$;

R_9: go to $M_3(2\delta)$; R_{10}: unload $M_3(\varepsilon)$;

R_{11}: go to output, drop the part, go to $M_3(\varepsilon+3\delta)$; R_{12}: unload $M_2(\varepsilon)$;

RP_j: wait at $M_j(w_j^i)$ s_i: starting time of R_i; sp_j: starting time of RP_j

α: M_1 is ready to be unloaded;

β: M_2 is ready to be unloaded;

γ: M_3 is ready to be unloaded;

By considering a multiple-part type system, at machine M_1, when we want to load a part on the machine we have to decide which part should be chosen such that the cycle time is

minimized. The same thing also can be achieved for M_2 and M_3. Based on the choosing gate definition we simply have three choosing gates as α, β, and γ. Thus we can write the following formulation using 0-1 integer variables $x1_{ij}$, $x2_{ij}$, and $x3_{ij}$ as:

$$\alpha_1 : s_{4,1} - s_{8,n} + C_t \geq \sum_{i=1}^{n} x1_{in}(a_i) + \varepsilon$$

$$\alpha_j : s_{4,j+1} - s_{8,j} \geq \sum_{i=1}^{n} x1_{ij}(a_i) + \varepsilon \qquad j = 2,...,n.$$

$$\beta_j : s_{12,j} - s_{6,j} \geq \sum_{i=1}^{n} x2_{ij}(b_i) + \varepsilon \qquad j = 1,...,n.$$

$$\gamma_j : s_{10,j} - s_{2,j} \geq \sum_{i=1}^{n} x3_{ij}(c_i) + \varepsilon \qquad j = 1,...,n.$$

Definition. A marked graph is a Petri-net such that every place has only one input and only one output.

Theorem 1. For a marked graph which every place has i m tokens (see figure 4), the following relation $s_B \geq s_A + m_i C_t$, where s_A, s_B are starting times of transitions *A* and *B* respectively, and C_t is cycle time, is true.

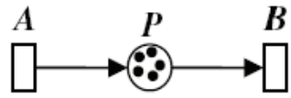


Fig. 4. The marked graph in theorem 1

Proof: see ref. (Maggot, 1984).

In addition the following feasibility constraints assign unique positioning for every job:

$$\sum_{i=1}^{n} x1_{ij} = 1 \quad j = 1,\cdots,n$$

$$\sum_{j=1}^{n} x1_{ij} = 1 \quad i = 1,\cdots,n.$$

To keep the sequence of the parts between the machines in a right order, we have to add the following constraints:

$$x1_{i,j} = x2_{i,j+1} \quad i = 1,...,n, j = 1,\cdots,n$$

$$x2_{i,j} = x3_{i,j+1} \quad i = 1,...,n, j = 1,\cdots,n.$$

Where, we assume that $x1_{i,n+1} = x1_{i,1}$ because of the cyclic repetition of parts.

Thus the complete model for the three machine robotic cell with multiple-part would be as follows:

$$\min C_t^6$$

Subject to:

$$p_{1,1} : s_{2,1} - s_{12,n} + C_t = \varepsilon + \delta \qquad j = 2,\cdots,n \qquad (1)$$

$$p_{1,j} : s_{2j} - s_{12j} = \varepsilon + \delta \qquad j = 1,\cdots,n \qquad (2)$$

$$p_{3,j} : s_{4j} - s_{2j} - w_{1j} = \varepsilon + 2\delta \qquad j = 1,\cdots,n \qquad (3)$$

$$p_{5,j} : s_{6j} - s_{4j} = \varepsilon + \delta \qquad j = 1,\cdots,n \qquad (4)$$

$$p_{7,j} : s_{8j} - s_{6j} = 2\varepsilon + 3\delta \qquad j = 1,\cdots,n \qquad (5)$$

$$p_{9,j} : s_{10j} - s_{8j} - w_{3j} = \varepsilon + 2\delta \qquad j = 1,\cdots,n \qquad (6)$$

$$p_{11,j} : s_{12j} - s_{10j} - w_{2j} = 2\varepsilon + 3\delta \qquad j = 1,\cdots,n \qquad (7)$$

$$\alpha_1 : s_{4,1} - s_{7,1} + C_t - \sum_{i=1}^{n} x1_{in}(a_i + \varphi_{a_i}) \geq \varepsilon \qquad (8)$$

$$\alpha_j : s_{4,j} - s_{7,j} - \sum_{i=1}^{n} x1_{ij}(a_i + \varphi_{a_i}) \geq \varepsilon \qquad j = 2,\cdots,n \qquad (9)$$

$$\beta_j : s_{12,j} - s_{6,j} - \sum_{i=1}^{n} x2_{ij}(b_i + \varphi_{b_i}) \geq \varepsilon \qquad j = 1,\cdots,n \qquad (10)$$

$$\gamma_j : s_{10,j} - s_{2,j} - \sum_{i=1}^{n} x3_{ij}(c_i + \varphi_{c_i}) \geq \varepsilon \qquad j = 1,\cdots,n \qquad (11)$$

$$x1_{i,j-1} = x2_{i,j} \qquad i,j = 1,\cdots,n \qquad (12)$$

$$x2_{i,j-1} = x3_{i,j} \qquad i,j = 1,\cdots,n \qquad (13)$$

$$\sum_{i=1}^{n} x1_{ij} = 1 \qquad j = 1,\cdots,n \qquad (14)$$

$$\sum_{j=1}^{n} x1_{ij} = 1 \qquad i = 1,\cdots,n \qquad (15)$$

$$s_{i,j} \geq 0, w_{kj} \geq 0, \; x1, x2, x3 \in \{0,1\}$$

4. The proposed hybrid particle swarm optimization (HPSO) algorithm

The particle swarm optimization (PSO) is a population based stochastic optimization technique that was developed by Kennedy and Eberhart in 1995 (Hu et al., 2004). The PSO inspired by social behavior of bird flocking or fish schooling. In PSO, each solution is a bird in the flock and is referred to as a particle. A particle is analogous to a chromosome in GAs (Kennedy and Eberhart, 1995). All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles.

The particles fly through the problem space by following the particles with the best solutions so far (Shi and Eberhart, 1998).
The general scheme of the proposed HPSO is presented in Figure 5.

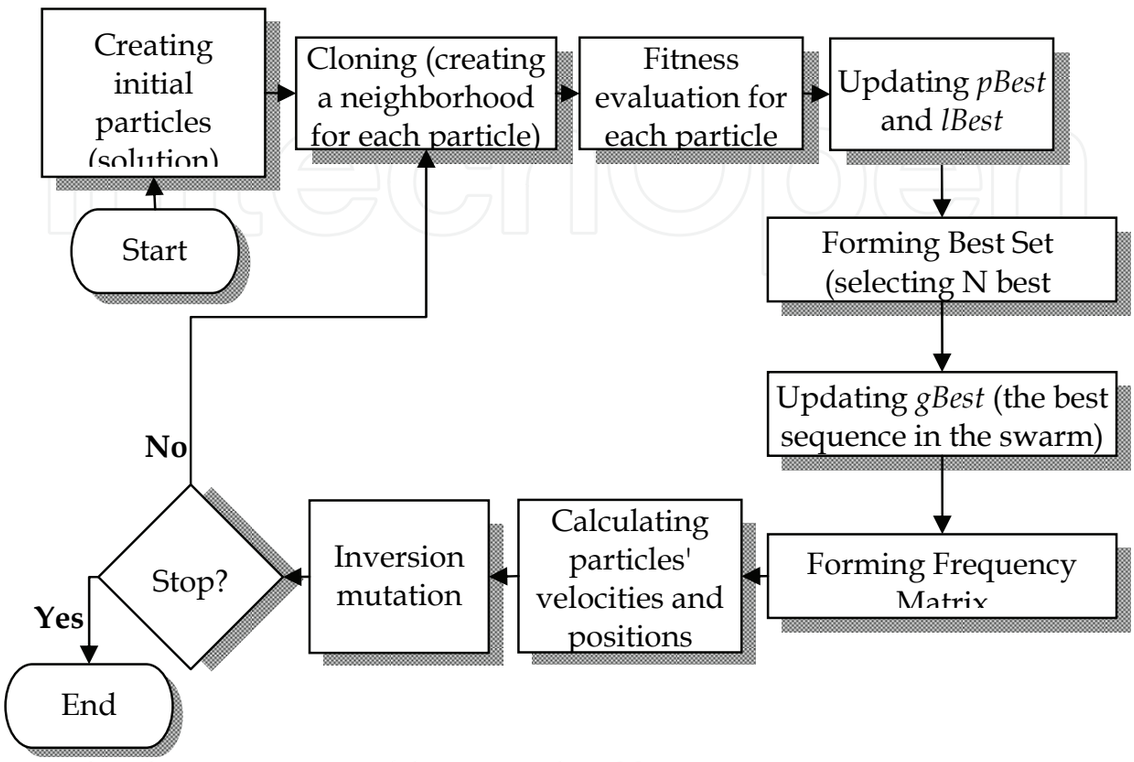


Fig 5. The schematic structure of the proposed HPSO

In this chapter, we extend the discrete PSO of Liao et al. (2007) to solve the robotic cell problem. In the proposed HPSO the velocity of each particle is calculated according to equation (16).

$$V_{id} = wV_{id} + c_1 \times rand() \times (pBest_{id} - x_{id}) + c_2 \times Rand() \times (lBest_{id} - x_{id}) - \left(t/\left(Number\ of\ Iterations\right)\right) \times \left(a \times Frequency\ Matrix - b\right) \quad (16)$$

Where c_1 and c_2 are the learning factors that control the influence of *pBest* and *lBest*. w is the inertia weight which controls the exploration and exploitation abilities of algorithm. $rand()$ and $Rand()$ are two independently generated random numbers, t is the current iteration and a and b are two parameters that adjust the influence of the Frequency Matrix on velocity value. *pBest* is the best position which each particle has found since the first step and it represents the experiential knowledge of a particle. After the cloning procedure (the detailed of cloning procedure will be described in the next section), a neighborhood for each particle is achieved. The best particle in this neighborhood is selected as *lBest*.

As Liao et al. (2007), the velocity values transfers from real numbers to the probability of changes by using the equation (17):

$$s(V_{id}) = 1/\left(1+\exp(-V_{id})\right) \tag{17}$$

where $s(V_{id})$ stands for the probability of x_{id} taking the value 1. In the proposed algorithm, the new position (sequence) of each particle is constructed based on its probability of changes that calculated by equation (17). Precisely, for calculating the new position of each particle, the algorithm starts with a null sequence and places an unscheduled job *j* in position *k (k = 1, 2, . . . , n)* according to the probability that determined by equation (18):

$$q_i(j,k) = s(V_{id}) \Big/ \sum_{j\in F} s(V_{id}) \tag{18}$$

where F is the set of the first f unscheduled jobs as present in the best particle (solution) obtained till current iteration. To achieve a complete sequence, the jobs are added one after another to the partial sequence.

The proposed HPSO terminates after a given number of iterations and the best sequence is reported as the final solution for the problem.

4.1 Cloning

For avoiding local optimal solutions we implement cloning procedure which in summary can be described as follows:

1. *M* copies of the solution are generated so that there are (*M*+1) identical solutions available.
2. Each of the *M* copies are subjected to the swapping mutation.
3. In each clone only the original solution participates in HPSO evolution procedure whereas the other copies of the solution would be discarded.
4. The above procedure is repeated for all of the solutions in the swarm.

4.2 Fitness evaluation

As any metaheuristic algorithm, the HPSO uses a fitness function to quantify the optimality of a particle (sequence). The cycle times of one unit for the policy s^6 are given by:

$$\mathrm{T}^{6}_{\mathrm{I},\sigma(\mathrm{i})\sigma(\mathrm{i+1})\sigma(\mathrm{i+2})} = 12\delta + 8\varepsilon + \max\{0, \mathrm{a}_{\sigma(\mathrm{i+2})} - 8\delta - 4\varepsilon, \mathrm{b}_{\sigma(\mathrm{i+1})} - 8\delta - 4\varepsilon, \mathrm{c}_{\sigma(\mathrm{i})} - 8\delta - 4\varepsilon\}$$

Hence, the following equeation is applied to calculate the fitness function.

4.3 Best Set formation

In the proposed HPSO, to improve efficiency of the algorithm, the best solutions which are obtained so far are selected and kept in the Best Set. Then, the Best Set is applied to forming the Frequency Matrix in next phase of the algorithm.

To form the Best Set, in the first iteration of algorithm and after the cloning phase of the algorithm, the *B* first best particles among all particles in the swarm are selected and placed

in the Best Set. In the other iterations, only the particles that better than the existed particles in the Best Set are replaced with them.

4.4 Frequency Matrix formation

The Frequency Matrix is a matrix which represents the average times that a specific job goes to a specific position according to sequence of particles in the Best Set. To illustrate the Frequency Matrix formation procedure, assume that the following particles are in the Best Set.

First particle (sequence): (1,2,3,4,5)
Second particle (sequence): (1,2,4,3,5)
Third particle (sequence): (1,2,3,5,4)

Therefore, the Best Set will be as follows (Figure 6):

Job \ Position	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	.66	.33	0
4	0	0	.33	.33	.33
5	0	0	0	.33	.66

Fig. 6. The example Frequency Matrix

4.5 Inversion mutation

The mutation operator causes a random movement in the search space that result in solution diversity. Inversion mutation is adopted in the proposed algorithm. The inversion mutation, as illustrated in Figure 7, selects two positions within a chromosome at random and then inverts the subsequence between these two positions.

1	**8**	**2**	**5**	**4**	**3**	**6**	**7**
1	**8**	**3**	**4**	**5**	**2**	**6**	**7**

Fig. 7. General scheme inversion mutation

5. Experimental Results

The performance of the proposed hybrid particle swarm optimization is compared with three well-known metaheuristic algorithms: GA, PSO-I, and PSO-II. These algorithms have been coded in the Visual Basic 6 and executed on a Pentium 4, 1.7 GHz, and Windows XP using 256 MB of RAM. Note that the performance of the proposed algorithm is also compared with Lingo 8 for small-sized problems.

5.1. Benchmark algorithms

At first, we present a brief discussion about the implementation of benchmark algorithms: GA, PSO-I, and PSO-II.

5.1.1 Genetic algorithm (GA)

Genetic Algorithm (GA) was developed by Holland in 1975 as a tool for solving complex optimization problems of large solution search spaces (Holland, 1992). GAs have been applied successfully to a wide variety of optimization problems to find optimal or near-optimal solutions (Gen and Cheng, 1997). Thus, for evaluating the performance and reliability of the proposed PSO algorithm, we use GA as one of three benchmark algorithms. A pseudocode for the applied GA is provided in Figure 8.

```
Begin;
  Generate random population of N solutions;
  For each solution: calculate fitness;
  For i=1 to number of generations (G);
    For j=1 to N × Crossovr_Rate;
      Select two parents randomly;
      Generate an offspring = crossover (Parent1 and Parent2);
      Calculate the fitness of the offspring;
      If the offspring is better than the worst solution then
        Replace the worst solution by offspring;
      Else generate a new random solution;
    Next;
    Do
      Copy the i^th best solution from previous generation to current generation;
    Until population size (N) is not reached;
    For k=1 to N × Mutation_Rate;
      Select one solution randomly;
      Generate a New_Solution = mutate (Solution);
    Next;
  Next;
End.
```

Fig. 8. Pseudocode for the Genetic Algorithm

5.1.2 PSO-I (Basic algorithm)

In this section, the structure of PSO-I (basic algorithm) is briefly described. The pseudocode of the applied PSO-I is provided in Figure 9.

```
Initialize the particle population randomly
Do
        Calculate fitness values of each particle
        Update pBest if the current fitness value is better than pBest
        Determine nBest for each particle: choose the particle with the best
        fitness value of all the neighbors as the nBest
        For each particle
                Calculate particle velocity according to (19)
                Update particle position according to (20)
While maximum iterations or minimum criteria is not attained
```

Fig. 9. Pseudocode for the PSO-I Algorithm (Shi and Eberhart, 1998)

PSO is initialized with a group of random particles and then search for optima by updating each generation. In each iteration, particles are updated by following two best values. The first one is the location of the best solution a particle has achieved so far which referred it as *pBest*. Another best value is the location of the best solution in all the population has achieved so far. This value is called *gBest* (Shi and Eberhart, 1998). Equation (19) calculates a new velocity for each particle as follows.

$$V_{id} = w \times V_{id} + c_1 \times Rand() \times (pBest_{id} - x_{id}) + c_2 \times rand() \times (nBest_{nd} - x_{id}) \quad (19)$$

Where $Rand()$ and $rand()$ are two random numbers independently generated. c_1 and c_2 are two learning factors, which control the influence of *pBest* and *nBest* on the search process. The global exploration and local exploitation abilities of particle swarm are balanced by using the inertia weight, w. Particles' velocities are bounded to a maximum velocity $V_{\max}$ for managing the global exploration ability of PSO (Shi and Eberhart, 1998). Equation (20) updates each particle's position (x_{id}) in the solution hyperspace.

$$x_{id} = x_{id} + V_{id} \quad (20)$$

5.1.3 PSO-II (Constriction algorithm)

In this section, the structure of PSO-II (constriction algorithm) is expressed in a few words. Also the structure of PSO-II is similar to PSO-I (as illustrated in Figure 4), but in PSO-II the velocity for each particle is calculated according to equation (21) (Engelbrecht, 2005).

$$V_{id} = \chi \left[V_{id} + \phi_1 \left(pBest_{id} - x_{id} \right) + \phi_2 \left(nBest_{nd} - x_{id} \right) \right] \quad (21)$$

Where

$$\chi = \frac{2k}{\left|2-\phi-\sqrt{\phi(\phi-4)}\right|} \tag{22}$$

With

$$\phi = \phi_1 + \phi_2 \tag{23}$$

$$\phi_1 = c_1 \times Rand() \tag{24}$$

$$\phi_2 = c_2 \times rand() \tag{25}$$

Equation (22) is employed by considering the constraints that $\phi \geq 4$ and $k \in [0,1]$. By employing the constriction approach under above mentioned constraints, convergence of the swarm to a stable point is guaranteed. The exploration and exploitation abilities of the algorithm are controlled by the parameter of equation (22): k (Engelbrecht, 2005).

Small-sized problem			Large-sized problem		
No. Of Parts	Problem Number	Problem Condition	No. Of Parts	Problem Number	Problem Condition
5	1	$a_i \geq b_i \geq c_i$	50	22	$a_i \geq b_i \geq c_i$
	2	$a_i \geq c_i \geq b_i$		23	$a_i \geq c_i \geq b_i$
	3	$b_i \geq a_i \geq c_i$		24	$b_i \geq a_i \geq c_i$
	4	$b_i \geq c_i \geq a_i$		25	$b_i \geq c_i \geq a_i$
	5	$c_i \geq a_i \geq b_i$		26	$c_i \geq a_i \geq b_i$
	6	$c_i \geq b_i \geq a_i$		27	$c_i \geq b_i \geq a_i$
	7	Unconditional case		28	Uncondition al case
10	8	$a_i \geq b_i \geq c_i$	75	29	$a_i \geq b_i \geq c_i$
	9	$a_i \geq c_i \geq b_i$		30	$a_i \geq c_i \geq b_i$
	10	$b_i \geq a_i \geq c_i$		31	$b_i \geq a_i \geq c_i$
	11	$b_i \geq c_i \geq a_i$		32	$b_i \geq c_i \geq a_i$
	12	$c_i \geq a_i \geq b_i$		33	$c_i \geq a_i \geq b_i$

	13	$c_i \geq b_i \geq a_i$		34	$c_i \geq b_i \geq a_i$
	14	Unconditional case		35	Unconditional case
15	15	$a_i \geq b_i \geq c_i$	100	36	$a_i \geq b_i \geq c_i$
	16	$a_i \geq c_i \geq b_i$		37	$a_i \geq c_i \geq b_i$
	17	$b_i \geq a_i \geq c_i$		38	$b_i \geq a_i \geq c_i$
	18	$b_i \geq c_i \geq a_i$		39	$b_i \geq c_i \geq a_i$
	19	$c_i \geq a_i \geq b_i$		40	$c_i \geq a_i \geq b_i$
	20	$c_i \geq b_i \geq a_i$		41	$c_i \geq b_i \geq a_i$
	21	Unconditional case		42	Unconditional case

Table 1. Problem inctances

5.2 Test Problems

To validate the proposed model and the proposed algorithm, various test problems are examined. The experiments are implemented in two folds: first, for small-sized problems, the other for large-sized ones. For both of these experiments, the values of ε and δ are equal to 1; the processing time for all parts on the all machine are uniformly generated in range [10, 100]. The problem instances are randomly generated as Table 1.

5.3 Parameters selection

For tuning the algorithms, extensive experiments were accomplished with different sets of parameters. In this section, we only summarize the most significant findings:

Genetic algorithm

No of Generation, Population Size, Crossover Rate (Linear order Crossover) and Mutation Rate (Inversion Mutation) for the small-sized problems were set to 50, 50, 1.0, and 0.2, respectively; and for the large-sized problems were set to 100, 100, 1.0 and 0.2, respectively.

PSO-I algorithm

No of Generation, Swarm Size, Learning factors (c_1 and c_2), and $V_{\max}$ for the small-sized problems were set to 50, 50, 2, 2, and 3, respectively; and for the large-sized problems were set to 100, 100, 2, 2, and 3. The inertia weight for all problem inctances was set to 1.4 that linearly decreases to 0.9 in each iteration.

PSO-II algorithm

5.4 Numerical results

In this section, the proposed HPSO is applied to the test problems, and its performance is compared with above mentioned benchmark algorithms. Each algorithm was executed for 15 times and the mean results were calculated. The numerical results for various test problems are presented in Tables 2 and 3.

Problem no.	Longo 8.0		GA			PSO-I			PSO-II		
	OFV[a]	Time	OFV		Time	OFV		Time	OFV		Time
			Ave.	STD		Ave.	STD		Ave.	STD	
1	483	<1	483	0	<1	483	0	<1	483	0	<1
2	435	<1	435	0	<1	435	0	<1	435	0	<1
3	363	<1	363	0	<1	363	0	<1	363	0	<1
4	459	<1	459	0	<1	459	0	<1	459	0	<1
5	454	<1	458	0	<1	458	0	<1	458	0	<1
6	404	<1	404	0	<1	404	0	<1	404	0	<1
7	321	<1	323	0	<1	323	0	<1	323	0	<1
8	754	1	754	0	<1	754.1	0.3	1	754	0	<1
9	763	1	763	0	<1	763	0	1	763	0	<1
10	910	<1	910	0	<1	910	0	1	910	0	<1
11	825	1	825	0	<1	825	0	1	825	0	<1
12	907	<1	907	0	<1	907	0	1	907	0	<1
13	753	<1	753	0	<1	753	0	1	753	0	<1
14	739	132	741.9	1.9	<1	746.5	6	1	744.4	6.1	<1
15	1312	<1	1312	0	1	1312	0	1	1312	0	<1
16	1272	<1	1272.1	0.3	1	1273.4	1.5	1	1274.2	1.9	<1
17	1212	1	1212	0	1	1212.7	0.6	1	1213.6	1.5	<1
18	1352	<1	1352	0	1	1352	0	1	1352	0	<1
19	1331	<1	1331	0	1	1331	0	1	1331	0	1
20	1222	1	1222	0	1	1226.7	4.4	1	1224	2.5	<1
21	1260	7200[c]	1145.9	18.9	1	1181.5	13.1	1	1178	13.9	<1

a Objective Function Value
b Standard Deviation
c denotes that the Lingo interrupted after this time and the best achieved value was reported

Table 2. Computational results for small-sized test problems

Problem no.	GA			PSO-I			PSO-II			HPSO		
	OFV		Time	OFV		Time	OFV		Time	OFV		Time
	Ave.	STD		Ave.	STD		Ave.	STD		Ave.	STD	
22	4414	0	12.6	4427.2	4.5	14.5	4424.6	6.0	14.2	4404.6	1.3	98
23	4227.1	0.3	12.2	4239.1	7.8	14.5	4238.4	7.4	14.1	4207	0	98.2
24	3997	0.2	12.5	4026.9	11.3	14.5	4026.8	10.7	14.2	3970.4	4.7	98.6
25	4314	0	13.1	4334.3	6.8	14.5	4332.6	5.2	14.2	4314	0	99.2
26	4283.6	2.3	12.1	4295.1	8.5	14.5	4289.8	3.6	14.2	4257.2	4.8	99.2
27	4360	0	13	4364.3	2.4	14.5	4364.4	2.4	14.1	4360	0	99
28	3405.4	37.1	11.2	3583.1	23.1	14.5	3615.4	26.2	14.2	3385.8	86.1	99
29	6287.8	1.2	19.5	6347	14.6	24.9	6345.4	13.7	24.8	6239.8	20.1	215.4
30	6360.3	0.7	19.3	6437.8	8.7	24.9	6446.2	13.8	24.5	6360.6	1.3	215.2
31	6369.1	0.5	19.9	6469.7	9.6	25.1	6471.4	13.5	24.5	6374.6	12.0	215.2
32	6375.5	0.7	19.5	6431.6	15.1	25.3	6446.4	10.0	24.5	6294.4	37.3	215.2
33	6716.1	1.9	19.3	6764.1	8.6	25	6761.4	7.3	24.8	6701.8	13.4	215.2
34	6357.1	3.9	19.9	6419.4	7.5	25.1	6423.6	10.8	24.8	6329.4	8.2	215.6
35	5843.6	45.1	18.7	6173.3	25	25.1	6181.8	35.4	24.5	5751.6	167.4	215.6
36	8832	0.7	29.5	8889.4	7.6	37.9	8887.8	16.3	37	8812.4	0.9	398.8
37	8693.5	7.3	28.1	8728.5	17	37.8	8747.8	11.8	35.9	8622.6	19.1	398.4
38	8735.9	3.5	27.8	8836.9	20.6	37.8	8842.2	17.0	35.8	8673.2	50.8	400
39	8663.3	2.8	28.7	8861.9	11.5	37.9	8674.6	17.6	36	8585.2	40.1	399.6
40	8125.8	3.9	27.8	8258.3	15.3	37.9	8296.4	11.9	34.3	8111.6	7.8	400.6
41	8588.1	0.3	29.6	8645.7	11.3	38.1	8654	14.6	35.3	8505.0	37.5	398.2
42	7837.9	75.7	27.8	8113.7	30.3	38.1	8163	36.9	36.1	7545.8	97.4	399.2

Table 3. Computational results for large-sized test problems

As shown in Tables 2 and 3, the proposed HPSO is superior to the benchmark algorithms in the most test problems.
As illustrated in Tables 2 and 3, the proposed HPSO consumes more computational time than the benchmark algorithms. Because of the structure of the proposed HPSO, it can search smartly more regions of the search space that results in better solutions. Thus, this higher value of computational time is reasonable.

6. Conclusions

This chapter developed a new mathematical model for a cyclic multiple-part type three-machine robotic cell problem under S^6 robot movement policy that minimizes the cycle time. The developed model is based on Petri nets and provides a new method to calculate cycle times by considering waiting times. It was proved that calculating cycle time under S^6 policy is unary NP-complete. Hence, this chapter proposed a new hybrid particle swarm optimization (HPSO) algorithm to tackle the problem. To validate the developed model and solution algorithm, various test problems with different sizes were randomly generated and the performance of the HPSO was compared with three benchmark metaheuristics: Genetic Algorithm, PSO-I (basic Particle Swarm Optimization algorithm), and PSO-II (constriction Particle Swarm Optimization algorithm). The numerical results showed that the proposed HPSO outperforms the benchmark algorithms in the most problems, especially for large-sized problems.

7. References

Agnetis, A. (2000). "Scheduling No-Wait Robotic Cells with Two and Three Machines." European Journal of Operational Research 123: 303-314.

Agnetis, A. and D. Pacciarelli (2000). "Part Sequencing in Three-Machine No-Wait Robotic Cells." Operations Research Letters 27: 185-192.

Asfahl, C. R. (1992). Robots and Manufacturing Automation. New York, Wiley.

Bagchi, T. P., J. N. D. Gupta and C. Sriskandarajah (2006). "A Review of Tsp Based Approaches for Flow Shop Scheduling." European Journal of Operational Research 169: 816- 854.

Brauner, N. and G. Finke (1999). "On a Conjecture About Robotic Cells: New Simplified Proof for the Threemachine Case." INFOR 37(1): 20-36.

Crama, Y., V. Kats, J. v. d. Klundert and E. Levner (2000). "Cyclic Scheduling in Robotic Flow Shops." Annals of Operations Research: Mathematics of Industrial Systems 96: 97-124.

Crama, y. and v. d. Klundert (1999). "Cyclic Scheduling in 3-Machine Robotic Flow Shops." Journal of Scheduling 2: 35-54.

Dawande, M., H. N. Geismar, S. P. Sethi and C. Sriskandarajah (2005). "Sequencing and Scheduling in Robotic Cells:Recent Developments." Journal of Scheduling 8: 387-426.

Drobouchevitch, I. G., S. P. Sethi and C. Sriskandarajah (2006). "Scheduling Dual Gripper Robotic Cell Oneunit Cycles." European Journal of Operational Research 171: 598-631.

Engelbrecht, A. P. (2005). Fundamentals of Computational Swarm Intelligence. South Africa, John Wiley & Sons, Ltd.

Gen, M. and R. Cheng (1997). Genetic Algorithms and Engineering Design. New York, Wiley.

Gultekin, H., M. S. Akturk and O. E. Karasan (2006). "Cyclic Scheduling of a 2-Machine Robotic Cell with Tooling Constraints." European Journal of Operational Research 174: 777–796.

Gultekin, H., M. S. Akturk and O. E. Karasan (2007). "Scheduling in a Three-Machine Robotic Flexible Manufacturing Cell." Computers & Operations Research 34: 2463 – 2477.

Hall, N. G., H. Kamoun and C. Sriskandarajah (1997). "Scheduling in Robotic Cells: Classification, Two and Three Machine Cells." Operations Research 45: 421-439.

Hall, N. G., H. Kamoun and C. Sriskandarajah (1998). "Scheduling in Robotic Cells: Complexity and Steady State Anhlysis." European Journal of Operational Research 109: 43-65.

Holland, J. H. (1992). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MA, MIT Press.

Hu, X., Y. Shi and R. Eberhart (2004). Recent Advances in Particle Swarm. Congress on Evolutionary Computation, CEC2004 IEEE.

Kennedy, J. and R. Eberhart (1995). Particle Swarm Optimization. Proceedings of the IEEE international conference on neural networks (Perth, Australia), NJ: IEEE Service Center.

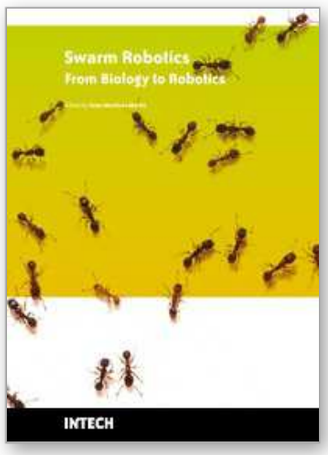
Liao, C.-J., C.-T. Tseng and P. Luarn (2007). "A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems." Computers & Operations Research 34(10): 3099-3111.

Maggot, J. (1984). "Performance Evaluation of Concurrent Systems Using Petri Nets." INFORM PROCESSING LETT 18(1): 7-13.

Sethi, S. P., C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak (1992). "Sequencing of Parts and Robot Moves in a Robotic Cell." International Journal of Flexible Manufacturing Systems 4: 331-358.

Shi, Y. and R. Eberhart (1998). A Modified Particle Swarm Optimizer. Proceedings of the IEEE international conference on evolutionary computation, Piscataway, NJ: IEEE Press.

Sriskandarajah, C., N. G. Hall, H. Kamoun and H. Wan (1998). "Scheduling Large Robotic Cells without Buffers." Annals of Operations Research: Mathematics of Industrial Systems 76: 287–321.



Swarm Robotics from Biology to Robotics

Edited by Ester Martinez Martin

ISBN 978-953-307-075-9
Hard cover, 102 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

In nature, it is possible to observe a cooperative behaviour in all animals, since, according to Charles Darwin's theory, every being, from ants to human beings, form groups in which most individuals work for the common good. However, although study of dozens of social species has been done for a century, details of how and why cooperation evolved remain to be worked out. Actually, cooperative behaviour has been studied from different points of view. Swarm robotics is a new approach that emerged on the field of artificial swarm intelligence, as well as the biological studies of insects (i.e. ants and other fields in nature) which coordinate their actions to accomplish tasks that are beyond the capabilities of a single individual. In particular, swarm robotics is focused on the coordination of decentralised, self-organised multi-robot systems in order to describe such a collective behaviour as a consequence of local interactions with one another and with their environment. This book has only provided a partial picture of the field of swarm robotics by focusing on practical applications. The global assessment of the contributions contained in this book is reasonably positive since they highlighted that it is necessary to adapt and remodel biological strategies to cope with the added complexity and problems that arise when robot individuals are considered.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Isa Nakhai Kamalabadi, Ali Hossein Mirzaei and Saeede Gholami (2010). A New Hybrid Particle Swarm Optimization Algorithm to the Cyclic Multiple-Part Type Three-Machine Robotic Cell Problem, Swarm Robotics from Biology to Robotics, Ester Martinez Martin (Ed.), ISBN: 978-953-307-075-9, InTech, Available from: http://www.intechopen.com/books/swarm-robotics-from-biology-to-robotics/a-new-hybrid-particle-swarm-optimization-algorithm-to-the-cyclic-multiple-part-type-three-machine-ro

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen